Transactions Briefs

Speedups and Energy Reductions From Mapping DSP Applications on an Embedded Reconfigurable System

Michalis D. Galanis, Gregory Dimitroulakos, and Costas E. Goutis

***Abstract*—This paper presents performance improvements and energy savings from mapping real-world benchmarks on an embedded single-chip platform that includes coarse-grained reconfigurable logic with a microprocessor. The reconfigurable hardware is a 2-D array of processing elements connected with a mesh-like network. Analytical results derived from mapping seven real-life digital signal processing applications, with the aid of an automated design flow, on six different instances of the system architecture are presented. Significant overall application speedups relative to an all-software solution, ranging from 1.81 to 3.99 are reported being close to theoretical speedup bounds. Additionally, the energy savings range from 43% to 71%. Finally, a comparison with a system coupling a microprocessor with a very long instruction word core shows that the microprocessor/coarse-grained reconfigurable array platform is more efficient in terms of performance and energy consumption.**

***Index Terms*—Coarse-grained reconfigurable array (CGRA), design flow, embedded systems, energy reduction, performance improvement, reconfigurable computing.**

I. Introduction

Reconfigurable architectures have received growing interest in the past few years [1]. Such systems usually combine reconfigurable hardware with a software-programmable general-purpose microprocessor. The microprocessor typically executes noncritical control intensive parts of the applications and provides software programmability. Compute-intensive sections, called *kernels*, are implemented in reconfigurable hardware. The operation parallelism present in kernels is exploited by the available abundant processing elements (PEs) of the reconfigurable hardware, resulting in performance improvements. Several coarse-grained reconfigurable architectures have been introduced [1]–[8]. Coarse-grained reconfigurable logic has been proposed as coprocessor for speeding-up kernels of multimedia and digital signal processing (DSP) applications in embedded systems. This type of hardware consists of PEs with word-level data bit-widths [like 32-bit arithmetic logic units (ALUs)] connected with a reconfigurable interconnect network. The coarse-grained PEs better exploit the word-level parallelism of many DSP applications than the field-programmable gate arrays (FPGAs) which are more effective for bit-level operations. The coarse granularity greatly reduces the delay, area, power consumption, and reconfiguration time relative to an FPGA device at the expense of flexibility [1], [2]. In this paper, we consider the most widespread subclass of coarse-grained architectures where the PEs are organized in a 2-D array and they are connected with mesh-like reconfigurable networks [1], [3]. In this paper, these architectures are called coarse-grained reconfigurable arrays (CGRAs).

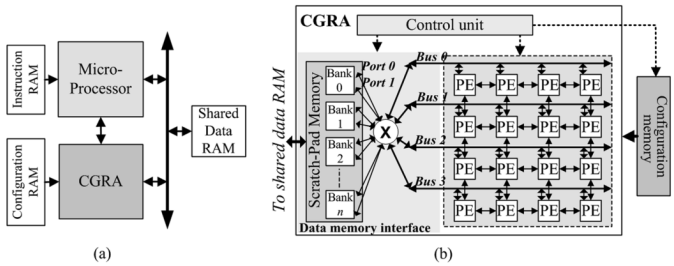

Fig. 1. (a) Outline of the embedded reconfigurable SoC platform. (b) CGRA architecture overview.

The main contribution of this paper is the extensive study of the performance improvements as well as the energy reductions by executing time critical kernels on the coarse-grained reconfigurable logic. A generic reconfigurable system coupling a microprocessor with a CGRA is the target platform. Seven real-world data dominated DSP benchmarks are mapped on three systems employing 32-bit ARM processors. Every ARM system is coupled each time with two different CGRA architectures, a 4×4 and a 6×6 array of PEs. The applications' execution times are evaluated using an automated design flow. The flow is proposed for mapping complete applications described in C software on the system platform. The core of the design flow is based on an efficient software pipelining algorithm that was introduced in [6]. Large values of instructions per cycles (IPCs) are achieved by the considered software pipelining-based mapper.

Few works exist for the thorough study on CGRAs. Research activities that consider the mapping of complete applications on processor/CGRA systems are presented in [3], [5], [7], and [8]. Our paper is more comprehensive than existing ones since more applications are mapped in the system, experiments with different instances of the system are performed and a high-level automated design flow that employs an efficient mapper for CGRAs is used. Furthermore, we present energy measurements at the system level whereas only one work [8] illustrated such type of measurements. Additionally, the comparison in executing complete applications with a microprocessor/very long instruction word (VLIW) system, which is a type of system widely used nowadays in embedded systems, was not performed in previous works. So, we consider that our study better reflects the computational abilities of CGRA systems.

The rest of this paper is organized as follows. Section II overviews the system architecture and the design flow. Section III presents the experiments, while Section IV concludes this paper.

II. System Architecture—Design Flow

A. Architecture Overview

Fig. 1(a) shows an overview of the reconfigurable system-on-chip (SoC) architecture considered in this paper. The platform is composed by a CGRA, an embedded microprocessor, instruction memory for storing program code, configuration memory for storing the complete

Manuscript received April 5, 2006; revised October 19, 2007 and February 26, 2007. This work was supported by the Alexander S. Onassis Public Benefit Foundation.

M. D. Galanis is with the VLSI Design Laboratory, Department of Electrical and Computer Engineering, University of Patras, Patras 26500, Greece (e-mail: mgalanis@ece.upatras.gr).

G. Dimitroulakos and C. E. Goutis are with the VLSI Design Laboratory, Department of Electrical and Computer Engineering, University of Patras, Patras 26500, Greece (e-mail: mgalanis@ece.upatras.gr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2007.909812

configuration of the CGRA, and shared data RAM. Local data and instruction (configuration) memories are located in both the microprocessor and in the CGRA. The execution model of the reconfigurable platform considers that the data communication between the CGRA and the microprocessor uses shared-memory mechanism. The shared memory is comprised by the system's shared data RAM and CGRA's local data memory which a scratch-pad one. Scalar variables, either live-in or live-out ones, are exchanged via the CGRA's scratch-pad. Global variables and data arrays are allocated in the system's shared data RAM. The communication process used by the microprocessor and the CGRA preserves data coherency by requiring the execution of the processor and the CGRA to be mutually exclusive.

For the CGRA, a flexible template architecture is considered which allows exploration in respect to various parameters. The CGRA template can represent a diversity of existing reconfigurable array architectures [1], [3] by being parametric in respect to: 1) the number and type (functionality) of PEs; 2) the interconnect structure among the PEs; and 3) their interface to the data memory. An overview of the considered CGRA template is shown in Fig. 1(b). A scratch-pad memory serves as a local data RAM for quickly loading data in the PEs. The PEs residing in a row or column share a common bus connection to the scratch-pad memory. The scratch-pad memory is composed by multiple banks for providing the necessary bandwidth to the CGRA PEs. Each bank's ports can be shared among the CGRA buses according to the description of the CGRA template. A control unit manages the execution of the CGRA every cycle by defining the operations performed by the PEs and the loading/storing of data from/to the memory. The microprocessor sets the control unit at the beginning of the kernel execution on the CGRA. A PE contains one functional unit (FU), which it can be configured to perform a specific word-level operation each cycle. Characteristic operations supported by the FU are ALU, multiplication, and shifts. For storing intermediate values between computations and data fetched from memory, local register file exists inside a PE. For more details about the CGRA template, the reader is referred to [6].

B. Design Flow Outline

Fig. 2 illustrates the diagram of the design flow for executing applications on the microprocessor/CGRA system. The input is an application described in the C language. Initially, a profiling procedure outputs the kernels and the noncritical parts of the source code. Kernels are considered those loops that contribute more than a certain amount to the total application's execution time on the processor. For example, loops that account 10% or more to the application's time can be characterized as kernels. The kernels are moved for execution on the CGRA. The intermediate representation (IR) of the kernel is created by utilizing the SUIF2 and MachineSUIF compiler infrastructures. Optimizations are then applied to the kernel's IR for efficient mapping after taking into account the CGRA characteristics, like the number of PEs in the CGRA. Examples of optimizations are dead code elimination, common subexpression elimination, constant propagation, if-conversion, and loop transformations. Transformations typically applied are loop unrolling and loop normalization. For instance, the loop unrolling is utilized for increasing the kernel's operation parallelism.

The optimized kernels are mapped on the CGRA for improving performance by utilizing our mapping procedure introduced in [6]. The second input to the mapper is a text file containing the description of the CGRA. The considered mapper [6] exploits instruction level parallelism in kernels by modulo scheduling which is a widely utilized software loop pipelining technique. The developed mapper advances existing works considering software pipelining approaches for CGRAs since: 1) concurrently encounters the scheduling, register allocation, and register spilling phases as these are highly related and 2) reduces the data bandwidth bottleneck by exploiting data reuse opportunities and by utilizing the PE's local storage and interconnections. For a detailed description of the mapper, the reader is refereed to [6]. The noncritical source code is compiled using a compiler for the specific processor. The performance is estimated via cycle-accurate simulation having as inputs the software binary of the processor and the configuration of the CGRA.

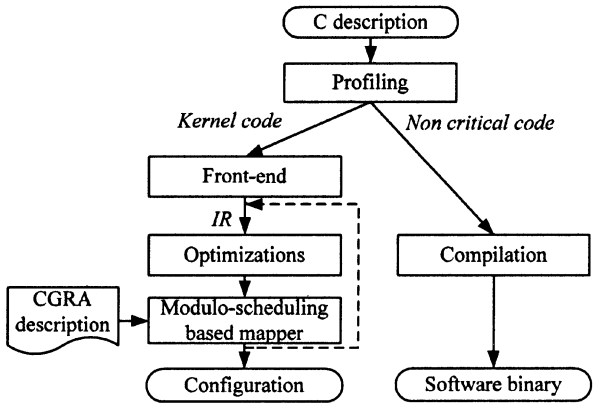

Fig. 2. Design flow for the microprocessor/CGRA platform.

III. Experiments

A. Set-Up

Two different CGRA architectures are used each time for accelerating critical kernels. The first architecture is a 4×4 array of PEs (*CGRA1*), while the second one (*CGRA2*) consists of 36 PEs connected in a 6×6 array. Both architectures have a data-width of 32-bits and their PEs are directly connected to all other PEs in the same row and in the same column through vertical and horizontal interconnections. The FU in each PE can execute a supported operation in one clock cycle. Two bidirectional buses per row are dedicated for transferring data to the PEs from the scratch-pad memory. Each bus transfers one 32-bit word with one-cycle delay. The CGRA architectures were described in register-transfer level VHDL. For the SRAMs, optimized components from TSMC 0.13-μm SRAM generators of artisan components were used. The Synopsys synthesis and power estimation tools were utilized to obtain delay, area, and power estimates for a 0.13-μm TSMC standard cell CMOS process from artisan components. It was found that the critical path delay allows the clock frequency to be 150 MHz for both CGRAs. The average power consumption for the 4×4 CGRA is 154.5 mW at 150 MHz, while for the 6×6 CGRA it is 258.0 mW at 150 MHz.

Three different architectures of 32-bit ARM processors are coupled each time with a CGRA. These processors are: 1) an ARM7 clocked at 133 MHz; 2) an ARM9 clocked at 250 MHz; and 3) an ARM10 having clock frequency of 325 MHz. The power consumption, at 0.13 μm for the ARM7 is 26.6 mW at 133 MHz, for the ARM9 is 112.5 mW at 250 MHz, while for the ARM10 is 195.0 mW at 325 MHz.

Seven real-world DSP applications, described in C language, are used in the experiments. These are a JPEG encoder, an IEEE 802.11 an OFDM transmitter, a wavelet-based image compressor, a medical imaging application called cavity detector, an image edge detection technique, a JPEG decoder, and a GSM voice encoder.

TABLE I
RESULTS FROM MAPPING KERNELS ON THE 4 × 4 CGRA

Kernel	# ops	MII	II	IPC	CGRA util. %
Jpeg_enc.k1	100	7	7	14.5	90.6
Jpeg_enc.k2	104	7	7	14.3	89.4
Jpeg_enc.k3	12	1	1	12.4	77.5
Jpeg_enc.k4	16	1	1	16.0	100.0
Ofdm_trans.k1	16	1	2	9.0	56.3
Ofdm_trans.k2	30	2	2	15.2	95.0
Ofdm_trans.k3	43	3	4	11.2	70.0
Ofdm_trans.k4	8	1	1	9.0	56.3
Compressor.k1	32	2	2	16.0	100.0
Compressor.k2	16	1	1	16.0	100.0
Compressor.k3	32	2	2	16.0	100.0
Compressor.k4	16	1	1	16.0	100.0
Cavity_det.k1	24	2	2	12.4	77.5
Cavity_det.k2	8	1	1	8.8	55.0
Cavity_det.k3	24	2	2	12.6	78.8
Cavity_det.k4	24	2	2	12.6	78.8
Edge_det.k1	18	2	3	7.0	43.8
Jpeg_dec.k1	70	5	5	15.1	94.4
Jpeg_dec.k2	48	3	3	16.0	100.0
Jpeg_dec.k3	62	4	4	14.8	92.5
Jpeg_dec.k4	80	5	5	16.0	100.0
Gsm_enc.k1	16	1	2	9.0	56.3
Gsm_enc.k2	16	1	1	16.0	100.0
Average				**13.3**	**83.1**

B. Speedups

1) Kernel Mapping Results: The results from mapping the critical loops of the applications on the 4 × 4 CGRA are given in Table I. The first column refers to the kernel *app.kn* of an application *app*, while the second one refers to the number of operations composing each loop after the unrolling performed for achieving better CGRA utilization and, consequently, better performance. The *MII* is the minimum initiation interval, while *II* is the interval actually achieved during modulo scheduling. The instructions per cycle (*IPC*) indicates the average number of computing operations executed per clock cycle in the scheduled loop. The IPC is a measure of the operation parallelism exploited and dictates the performance in modulo schedulers. The CGRA utilization is the average percentage of the PEs active per cycle and it is equal to the IPC divided by the number of PEs in the CGRA.

From Table I, it is inferred that the achieved *II* is equal to the *MII* in 19 out 23 kernels, a fact that reflects the quality of the CGRA mapping. The 4 × 4 CGRA is efficiently utilized since the average percentage of the CGRA utilization is 83.1%. The large values of CGRA utilization reveal the high-performance mapping of the kernels on the 4 × 4 array.

The maximum combined *II* (sum of the *II* values for an application's loops) among the applications equals 17 and refers to the JPEG decoder. The *II* defines the number of configuration words needed to execute the loop. Since in our experimental scenario each PE's local configuration RAM stores 32 contexts, there will be no time overhead for loading the local context RAMs during the execution of each application as all the PE configurations for the application's kernels can be stored in the local context RAM. When the kernels are mapped on the 6 × 6 CGRA, the achieved average CGRA utilization is 71.7%.

2) Application Speedups: The execution times and the application speedups for the seven applications are presented in Table II. Time_{sw} represents the software execution time of the whole application on a specific microprocessor (Proc.). The ideal speedup (*Ideal Sp.*) is the application speedup that would ideally be achieved, according to Amdahl's Law, if application's kernels were executed on the CGRA in zero time. $\text{Time}_{\text{system}}$ corresponds to the execution time of the application when the critical code is executed on the CGRA, either the CGRA1 or the CGRA2. All execution times are normalized to the software execution times on the ARM7. The *Sp.* is the estimated application speedup, after utilizing the developed design flow, over the execution of the application on the microprocessor.

TABLE II
COMPARISON OF EXECUTION TIMES FOR MICROPROCESSOR EXECUTION AND MICROPROCESSOR WITH CGRA

Application	Proc.	Time_{sw}	Ideal sp.	Proc./CGRA1		Proc./CGRA2	
				$\text{Time}_{\text{system}}$	Sp.	$\text{Time}_{\text{system}}$	Sp.
JPEG enc.	ARM7	1.000	3.96	0.270	3.71	0.266	3.76
	ARM9	0.461	3.24	0.155	2.98	0.151	3.05
	ARM10	0.301	3.16	0.109	2.75	0.106	2.84
OFDM trans.	ARM7	1.000	3.54	0.308	3.25	0.301	3.32
	ARM9	0.485	3.43	0.157	3.09	0.152	3.19
	ARM10	0.344	3.23	0.122	2.82	0.117	2.94
Compressor	ARM7	1.000	2.51	0.444	2.25	0.429	2.33
	ARM9	0.424	2.32	0.203	2.09	0.193	2.20
	ARM10	0.283	2.21	0.153	1.85	0.138	2.05
Cavity det.	ARM7	1.000	2.38	0.488	2.05	0.457	2.19
	ARM9	0.480	2.29	0.251	1.91	0.229	2.10
	ARM10	0.355	2.17	0.196	1.81	0.181	1.96
Edge det.	ARM7	1.000	2.61	0.413	2.42	0.403	2.48
	ARM9	0.498	2.54	0.217	2.30	0.208	2.39
	ARM10	0.367	2.49	0.166	2.21	0.158	2.32
JPEG dec.	ARM7	1.000	4.17	0.253	3.95	0.251	3.99
	ARM9	0.418	3.85	0.114	3.68	0.113	3.71
	ARM10	0.273	3.64	0.079	3.47	0.078	3.52
Gsm enc.	ARM7	1.000	3.05	0.348	2.87	0.342	2.92
	ARM9	0.426	2.93	0.153	2.78	0.151	2.83
	ARM10	0.295	2.88	0.108	2.72	0.106	2.77
Average			**2.98**		**2.71**		**2.80**

From the results given in Table II, it is evident that significant overall performance improvements are achieved when critical software parts are mapped on the reconfigurable logic. These speedups range from 1.81 to 3.99. It is noticed from Table II that the largest application performance gains are achieved for the ARM7-based systems since the ARM7 exhibits the highest cycles per instruction (CPI) and it has the slowest clock relative to the rest two ARM processors. The average application speedup of the data dominated DSP benchmarks for the ARM7-based systems is 2.96, for the ARM9 systems is 2.74, while for the ARM10 systems is 2.57. For the case of mapping the kernels on the 6 × 6 CGRA, the application speedups are somewhat larger than the 4 × 4 CGRA case. The larger application speedups are due to the better kernel speedups that obtained with the 6 × 6 array relative to the 4 × 4 CGRA. The overall application speedup slightly increases due to the fact that the noncritical code segments are executed on the microprocessor. The average estimated speedup is 2.71 for the 4 × 4 CGRA systems, while for the 6 × 6 CGRA ones equals to 2.80.

We notice that the reported estimated speedups for each application and for each processor type are somewhat close to the ideal speedups determined by the Amdahl's Law, especially for the case of the 6 × 6 CGRA systems. In particular, the average estimated speedup for the CGRA2 is 6.0% smaller than the average ideal speedup for all the processor systems. This illustrates that the proposed design flow efficiently exploited the processing capabilities of the CGRAs for speeding-up the applications close to the theoretical bounds.

C. Energy Savings

The energy savings by utilizing a CGRA in the microprocessor systems are presented. For estimating the total energy of the system the following formula is used:

$$E_{\text{total}} = \text{Time}_{\text{proc}} \cdot (P_{\text{proc}} + 0.20 \cdot P_{\text{CGRA}} + P_{\text{mem\_icon}}) + \text{Time}_{\text{CGRA}} \cdot (P_{CGRA} + 0.25 \cdot P_{\text{proc}} + P_{\text{mem\_icon}}) \quad (1)$$

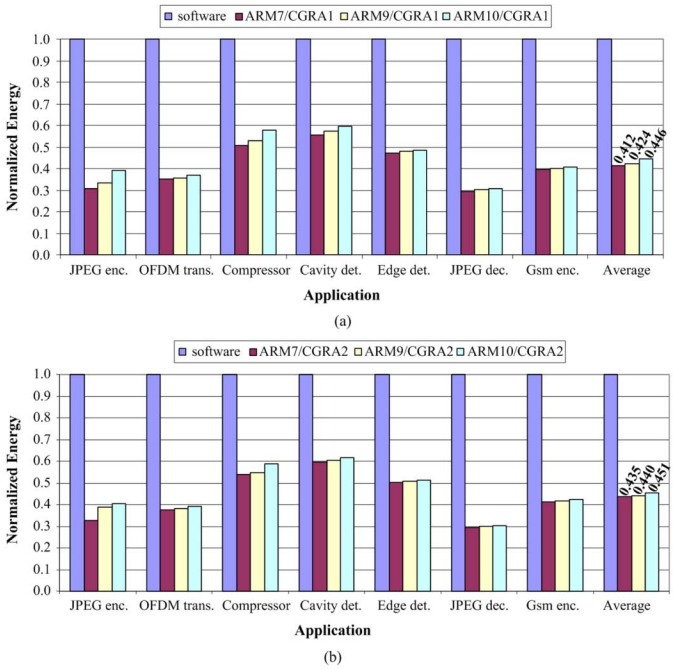

Fig. 3. Normalized energy consumption for the (a) 4 × 4 CGRA and (b) for the 6 × 6 CGRA systems.

where $\text{Time}_{\text{proc}}$ is the time of noncritical software parts on the microprocessor, $\text{Time}_{\text{CGRA}}$ is the execution time of the kernels, P_{proc} is the active power of the microprocessor, P_{CGRA} is the active power of the CGRA, and $P_{\text{mem\_icon}}$ is the power consumption of the shared data RAM and of the interconnection. The active power is when the logic (either microprocessor or the CGRA) evaluates. In (1) it is considered that the low-power idle mode (standby) of the CGRA consumes 20% of the power of its active state as in standby mode the power is primarily due to current leakage. In contemporary CMOS processes, leakage can account for 10% to 30% of the active power. The power-down mode of the microprocessor dissipates 25% of its power when it is active. It is assumed that the systems have $P_{\text{mem\_icon}}$ of 210 mW. This value was estimated from commercial and academic microprocessor platforms implemented at 0.13 μm as well as from industrial SRAM modules.

Fig. 3(a) illustrates the normalized energy consumptions when the 4 × 4 CGRA is coupled with each one of the three ARM processors. The normalized energy values for the 6 × 6 CGRA systems are presented in Fig. 3(b). The energy values are normalized to the software-only execution of each application on the microprocessor. From the presented results it is inferred that significant energy savings are achieved by executing critical kernels on the CGRAs. The largest energy reductions for both CGRA systems are reported for the JPEG decoder. The energy is reduced by an average of 59% for the ARM7/CGRA1 systems. The savings are slightly smaller for the ARM9 and ARM10-based platforms due to the smaller application speedups relative to the ARM7 systems as these are shown in Table II. More specifically, the system energy is smaller by an average of 58% relative to the all-software solution for the ARM9/CGRA1 systems, while for the ARM10/CGRA1 systems the energy is reduced by 55%. The energy savings for the ARM/CGRA2 platforms, for all applications, are slightly smaller relative to the ARM/CGRA1 systems. This is due to the larger power dissipation of the 6 × 6 CGRA relative to the 4 × 4 CGRA and due to the slightly larger application speedups that cannot result in further reduction of the energy consumption relative to the all-software solution.

D. Comparison With a VLIW-Based Microprocessor System

In this section, we present a comparison in terms of performance and energy consumption when complete applications are mapped on microprocessor/CGRA and on microprocessor/VLIW systems. The CGRA used is the 4 × 4 as achieves better energy savings than the 6 × 6 CGRA systems. A unified VLIW processor with eight functional units is coupled each time with an ARM processor. Eight-issue VLIWs are widely used as coprocessors in system-on-chips. For example, the commercial Digital Media SoC TMS320DM6446 of Texas Instruments couples an ARM9 and a TMS320C64x DSP VLIW core. The Trimaran framework is used for defining the architecture of the eight-issue VLIW processor, for compiling the kernels on the VLIW and for obtaining their execution cycles. The VLIW has the same operating frequency as the CGRA. We have selected the same clock period for having a straightforward comparison of the computational abilities (and of the mapping algorithms of the CGRA and the VLIW) without being affected by a different clock period. We mention that a considerably larger clock frequency is expected to be achieved for the CGRAs if they are designed in custom logic instead of standard cell logic as in this paper.

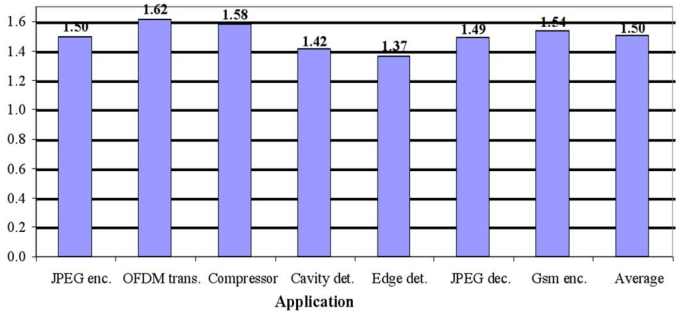

Fig. 4. Kernel acceleration achieved by the 4 × 4 CGRA relative to an eight-issue VLIW.

TABLE III
APPLICATION SPEEDUP AND ENERGY SAVINGS COMPARISON FOR THE ARM/CGRA AND THE ARM/VLIW SYSTEMS

Application	Proc.	App. Speedup		Energy Savings %	
		CGRA	VLIW	CGRA	VLIW
JPEG enc.	ARM7	3.71	3.44	69.3	66.4
	ARM9	2.98	2.77	66.7	64.1
	ARM10	2.75	2.58	61.0	58.7
OFDM trans.	ARM7	3.25	3.02	64.9	62.2
	ARM9	3.09	2.89	64.3	61.9
	ARM10	2.82	2.65	63.2	61.2
Compressor	ARM7	2.25	2.10	49.3	47.3
	ARM9	2.09	1.95	47.3	45.5
	ARM10	1.85	1.73	42.0	40.6
Cavity det.	ARM7	2.05	1.91	44.6	42.8
	ARM9	1.91	1.79	42.5	41.0
	ARM10	1.81	1.70	40.5	39.1
Edge det.	ARM7	2.42	2.26	52.8	50.7
	ARM9	2.30	2.14	52.0	50.1
	ARM10	2.21	2.08	51.5	49.8
JPEG dec.	ARM7	3.95	3.68	70.8	67.9
	ARM9	3.68	3.43	69.9	67.3
	ARM10	3.47	3.26	69.3	67.2
Gsm enc.	ARM7	2.87	2.65	60.3	57.3
	ARM9	2.78	2.59	59.7	57.2
	ARM10	2.72	2.56	59.4	57.3
Average		**2.71**	**2.53**	**57.2**	**55.0**

Fig. 4 shows the acceleration that the 4 × 4 CGRA provides to the kernels in respect to the considered VLIW. It is deduced that the CGRA requires 1.50 times less cycles on average for executing the kernels of an application. Thus, the CGRA mapper efficiently exploits the larger number of processing elements in the CGRA leading in important cycles reduction over a VLIW processor.

Table III presents the application speedups and the energy savings for the ARM/CGRA and the ARM/VLIW systems. The speedups and

the energy savings are relative to the all-software execution. The VLIW processor is assumed to consume 1.00 mW/MHz at 1.2 V which is the typical case in the commercial VLIW family TMS320C64x of Texas Instruments. So, the VLIW dissipates 150 mW at 150 MHz. For estimating the energy consumption of the VLIW-based systems, (1) is utilized. It is assumed that the low-power idle mode (standby) of the VLIW consumes 20% of the power of its active state which is also the case for the 4 × 4 CGRA.

From the results shown in Table III, it is noticed that the 4 × 4 CGRA always achieves better overall application speedups than the VLIW systems. This is due to the kernel acceleration achieved by the 4 × 4 CGRA relative to the VLIW. The average value of the speedups for the specific benchmarks is 2.71 for the 4 × 4 CGRA systems, while the VLIW platforms achieve an average speedup of 2.53. These performance advancements for the CGRA systems lead to slightly larger energy savings relative to the VLIW platforms. The area of a representative VLIW DSP core (the TMS320C64x) is about 12 mm^2 for a semi-custom standard cell implementation. The standard cell implementation of the 4 × 4 CGRA occupies 9.4 mm^2 of silicon. So, we can deduce that it is better to couple the 4 × 4 CGRA with a microprocessor as it is more efficient in terms of performance, energy consumption, and area relative to a processor/VLIW system.

IV. Conclusion

Speedups and energy reductions by executing realistic applications on six instances of a generic system architecture were presented. Significant speedups and energy reductions were achieved, while a comparison with a system containing a VLIW shows that the CGRA platforms achieve better speedups and comparable energy savings. Speedups and energy reductions from mapping DSP applications on an embedded reconfigurable system.

References

[1] T. J. Todman, G. A. Constantinides, S. J. E. Wilton, O. Mencer, W. Luk, and P. Y. K. Cheung, "Reconfigurable computing: Architectures and design methods," *IEE Comput. Digit. Tech.*, vol. 152, no. 2, pp. 193–207, Mar. 2005.

[2] C. Ebeling, C. Fisher, G. Xing, M. Shen, and H. Liu, "Implementing an OFDM receiver on the RaPid reconfigurable architecture," *IEEE Trans. Comput.*, vol. 53, no. 11, pp. 1438–1448, Nov. 2004.

[3] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. C. Filho, "MorphoSys: An integrated reconfigurable system for data-parallel and communication-intensive applications," *IEEE Trans. Comput.*, vol. 49, no. 5, pp. 465–481, May 2000.

[4] J. Becker and A. Thomas, "Scalable processor instruction set extension," *IEEE Des. Test Comput.*, vol. 22, no. 2, pp. 136–148, 2005.

[5] B. Mei, F.-J. Veredas, and B. Masschelein, "Mapping an H.264/AVC decoder onto the ADRES reconfigurable architecture," *Proc. FPL*, pp. 622–625, 2005.

[6] G. Dimitroulakos, M. D. Galanis, and C. E. Goutis, "Exploring the design space of an optimized compiler approach for mesh-like coarse-grained reconfigurable architectures," in *Proc. IPDPS*, 2006, pp. 1–10.

[7] J. Lee, K. Choi, and N. D. Dutt, "Evaluating memory architectures for media applications on coarse-grained reconfigurable architectures," *Proc. IEEE ASAP*, pp. 172–182, 2003.

[8] F. Barat, M. Jayapala, T. Vander Aa, R. Lauwereins, G. Deconinck, and H. Corporaal, "Low power coarse-grained reconfigurable instruction set processor," *Proc. FPL*, pp. 230–239, 2003.